

The Role of Conversation Policy in Carrying Out Agent Conversations

Laurence R. Phillips and Hamilton E. Link

Sandia National Laboratories

MS 0445

Albuquerque, NM 87185

lrphill@sandia.gov , helink@sandia.gov

Abstract. Structured conversation diagrams, or conversation specifications, allow agents to have predictable interactions and achieve predefined information-based goals, but they lack the flexibility needed to function robustly in an unpredictable environment. We propose a mechanism that dynamically combines conversation structures with separately established policies to generate conversations. Policies establish limitations, constraints, and requirements external to specific planned interaction and can be applied to broad sets of activity. Combining a separate policy with a conversation specification simplifies the specification of conversations and allows contextual issues to be dealt with more straightforwardly during agent communication. By following the conversation specification when possible and deferring to the policy in exceptional circumstances, an agent can function predictably under normal situations and still act rationally in abnormal situations. Different conversation policies applied to a given conversation specification can change the nature of the interaction without changing the specification.

1 Introduction

A: An argument is a connected series of statements intended to establish a proposition.

B: No, it isn't!

A: Yes, it is! It isn't just contradiction!

Policy discussion, Monty Python,

Argument Clinic Sketch

Software agents communicate while they pursue goals. In some cases, agents communicate specifically in order to accomplish goals. We restrict our interest in this paper to goals that can be described as information states, that is, *information goals*. We discuss agents that intend to accomplish information goals by communicating.

Although individual speech acts have been well-characterized, consensus on higher-order structured interactions has not been reached. There is little or no discussion in the literature of how to constrain the behavior of an agent during communication in response to a dynamic environment.

When a set of communication acts among two or more agents is specified as a unit, the set is called a *conversation*. Agents that intend to have a conversation require internal information structures that contain the results of deliberation about which communication acts to use, when to use them, whom the communications should address, what responses to expect, and what to do upon receiving the expected responses. We call these structures *conversation specifications*, or specifications for short. We claim that specifications are inadequate for fully describing agent behavior during interaction.

Consider two agents who are discussing the location of a surprise party for a third agent, who is not present. When that agent enters the room, all discussion of the party suddenly ceases. The cessation occurs because the first two agents understand that the third agent cannot receive any information that such a party is being considered. Conversely, suppose that the conversation is about a party in honor of the third agent and all three agents know the third agent is aware of it. Now, when the third agent enters the room, the conversation continues.

Are the first two agents having the same conversation in both cases? We claim the answer is “Yes, but they’re operating under different policies.” In both cases, they are having a conversation whose essence is organizing the party. The conversation might roughly be specified to contain information exchange components (e.g., to establish a set of possible locations), allocation components (“I’ll call these two restaurants, and you call this other one”), and a continuation-scheduling component (“I’ll call you tomorrow with what I find out and we’ll take it from there”). These are all matters that we expect to find in a conversation specification. On the other hand, the decision of whether to stop talking when a specific third party enters the room is based on a mutually understood policy and might reasonably be applied to any number of conversations, for example, negotiations about the price of a commodity on which the third agent is bidding.

Historically the agent communication literature has used the word “policy” to refer to the description of the structure of interaction between a number of agents, generally

two but sometimes more (Bradshaw et al. 1997). The dictionary, however, defines “policy” as “a high-level overall plan capturing general goals and acceptable procedures.” This coincides with what we expect of a conversation policy: an agent using a conversation *policy* would operate within certain constraints while attempting to satisfy general information-based goals. When discussing procedures and constraints of interaction beyond the basic structure of a conversation, the word “policy” has connotation that we feel is more appropriately bound to the procedures and constraints rather than to the basic structure. For the latter, then, we will instead use the word “specification,” and use the word “policy” to refer to the former.

2 Policies for Interaction

The focus of our work is to create a mechanism for combining specifications with policies that constrain the behavior of an agent in order to generate conversations among agents.

We have begun to design a mechanism that uses the specification’s description of input states and actions based on them and the policy’s description of constraints, limitations, and requirements together to determine an agent’s response to a message. Given a suitable mechanism, the specification and the policy can be implemented as data objects. The specification defines the structure for the conversation, and the policy defines the acceptable procedures, rules, and constraints for the conversation.

We can interact with and speak of agents as intentional systems (Dennett 1987). We assume that agents are able to emit illocutions and that illocutions can have perlocutionary effect on other agents that “hear” them (Searle 1969). (We follow Searle in using *illocution* to mean an utterance intended to affect the listener and *perlocution* to mean the production of effect on the listener). This means that an agent can emit information with the intent of altering the information state of some other agent, that the information can be received by some other agent, and that receipt of this information can cause the recipient to be in an information state intended by the emitter. The emitter desires the recipient to be in a certain state because the emitter believes that this either is or assists in achieving one or more of its goal states.

Conversation specifications are distinctly similar to KAoS conversation policies (Bradshaw et al. 1997). The specification dictates the transitions and outputs made by the agent in response to input. A conversation policy is a set of constraints on the conversation specification that limit the behavior of an agent beyond the requirement of following the procedures and structures of the conversation specification. The

policy object is used by the mechanism to make decisions about acceptable courses of action when the conversation specification fails to completely determine a course of action. Lynch and Tuttle said it well: “Our correctness conditions are often of the form ‘if the environment behaves correctly, then the automaton behaves correctly.’” (Lynch and Tuttle, 1989) This stems from the constraint that IOA’s cannot block inputs, the automaton is permitted to exhibit arbitrary behavior when “bad” or unexpected inputs occur. What happens when the environment *doesn’t* behave “correctly?” This is where policy applies.

Policy differs from specification in that specifications describe individual patterns of interactions, while policies are sets of high-level rules governing interactions. It is possible for a class of conversation policies to have subclasses. For one policy to be a subclass of another, the subclass must be more strict (more constraining) in at least one attribute and no less constraining in any.

Our new mechanism combines the policies and specifications to determine the set of conversations that can be generated. When policies change in the midst of a conversation, the goal may become infeasible. In our formulation, the conversation policy does not specify the types of messages that can occur. It is made up of constraints on who can participate, and under what circumstances, whether sub-conversations can be initiated within an existing open conversation, whether equivalent conversations can take place in parallel with the same participating entities (e.g., an agent can’t carry on two price negotiation conversations with the same entity w.r.t. the same object). We claim that issues of specification are orthogonal to issues of policy; specifications define the structure of interactions, while policies govern the way interactions are carried out.

3 Methods

We developed our current agent conversation mechanism using the Standard Agent Architecture (SAA) developed by the Advanced Information Systems Lab (Goldsmith, Phillips, and Spires 1998) at Sandia National Laboratories. The SAA provides a framework for developing goal-based reasoning agents, and we are currently using a distributed object system that enables agents to send each other simple objects or sets of information. We are using the Knowledge Query and Manipulation Language (KQML) (Labrou and Finin 1997) as our message protocol.

Interacting with an agent first requires that the agent be able to correctly identify and respond to illocutionary messages. A situated agent in pursuit of goals must be

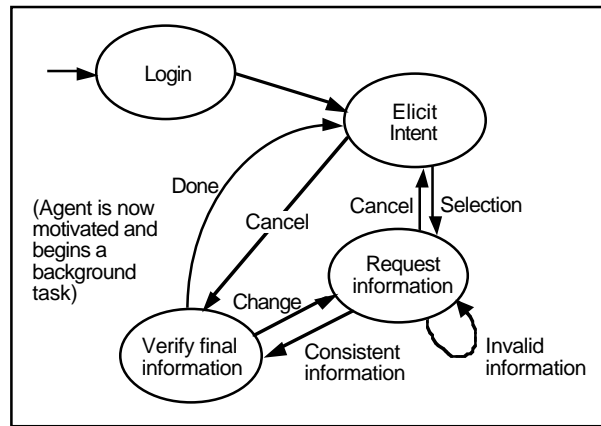


Fig. 1. A conversation specification that does not specify a variety of potential constraints on the agent's activities

able to answer two questions: To which, if any, of its current goals does new information relate, and what actions, if any, should it execute based on new information that relates to a given goal? In the SAA, the primary structure that enables this is the agent's stimulus-response table (SRT). An agent anticipating input of a certain type puts an entry into its SRT, which maps stimuli (by class or instance) to the appropriate action. Our system currently requires messages to contain an explicit reference to the context within which the SRT entry was created. The reference is realized as the object identifier (OID) of the current conversation object that gave rise to the message.

When an input arrives, the appropriate SRT entry is retrieved and its goal is undeferred (having previously been deferred, presumably awaiting relevant input), which activates the goal. The agent now determines how the new information in the context affects the goal and either marks it satisfied, failed, or deferred or continues to attempt to satisfy the goal. When satisfaction of the goal requires a speech act, the agent creates an utterance, delineates the context, embeds the context signature in the utterance, attaches the goal to the context, places the entry in the SRT, defers the goal, and executes the utterance. In short, illocution is a deliberate act that creates an utterance and sets up an expectation of the response that the recipient will make.

To engineer a conversation, the entire set of context descriptors of interest is laid out as a set of subgoals, each of which is satisfied by gathering specific information. We have automated the construction of an utterance from a context, the updating of the context to reflect the new information conveyed by the input, and the connectivity

that enables the utterance and the input to refer to the same context. Specialized code is written to construct goals, execute side effects, maintain the SRT, and so on.

Composing speech acts in a theoretically predictable fashion is more difficult; this is the motivation for creating a structured way of merging specification and policy at run time to get a structured interaction that is forced to remain within certain operational boundaries.

In our current mechanism, policy is embedded in the conversation mechanism as part of the design. A policy change, for example, that an agent should institute a timeout and ignore all messages responding to a particular request after the timeout expires, would require reengineering the conversation. The mechanism would be much more maintainable given an explicit policy object that could just be changed to reflect the fact that there's now a timeout. Our essential thesis is that policies and conversation specifications should be independent so that conversations could be switched under the same policy and policies could be changed without changing existing conversations.

4 Conversation policy

Consider the conversation in Figure 1. It describes a session allowing agent A to determine agent B's identity, offer B a choice of services and ascertain B's selection, and perform a task based on the selection. Describing the conversation is generally simple for such things: when a request or assertion comes in, the agent deliberates, returns information to the initiator, and anticipates the continuation. The two participants are responding to one another in turn, barring interruption, retransmission, or communication failure. There is no representation of what happens when the conversation is interrupted or when an agent retransmits a message. These issues are matters of policy that must be dealt with separately.

KAoS conversation "policies" enable definite courses of action to be established and fail-stop conditions to be dealt with (Bradshaw et al. 1997). They also imply mechanisms for initiating and concluding conversations. Specifications play the crucial role in agent communication of providing structure, but they do not, for example, describe whether a discussion can be postponed, or, if so, under what conditions or for how long. Indeed, KAoS conversation "policies" appear to concern matters of conversation *specification*, fundamentally how to respond to input given the current information state, rather than matters of conversation *policy*, such as what to

do when interrupted, whether the conversation can be postponed, or whether there is a time constraint on reaching an end state.

Policy issues are important. One constraint imposed by the policy in Figure 1 is that it requires turn-taking. If agent A receives several messages in a row, it may respond to each in turn without realizing that, say, B's third message was sent before A's second response. If agent A cannot detect the violation of the turn-taking policy, it might consider the second and third messages in an outdated context. A similar situation could occur if several agents were communicating and one were speaking out of turn. Without policy, designing a mechanism to deal with these violations means that a conversation specification that enforced turn-taking and one that merely allowed it would be two different things that would need to be maintained separately and activated separately by the agent. Furthermore, designing them into a system that had no notion of turn-taking would require that every state/action pair of every conversation specification be examined to see what should now happen if turn-taking is violated. At worst, accommodating a single policy issue doubles the number of conversation specifications an agent might be called upon to employ.

Examining constraints immediately leads to ideas for policies that replicate familiar patterns of interaction, such as a forum policy or a central-point-of-contact policy. Different classes of states, changes in context, and the particular protocol of communication used are independent of the conversation policy, although some make more sense in one policy or another. The web page and information-state context, for example, make the most sense in a 1:1 turn-taking policy when dealing one-on-one with a number of individual humans. KQML, in contrast, has many performatives that support broadcasting to a group of agents involved in the same conversation. In practical terms we may end up having to constrain which policies can be upheld based on communication details.

An explicit representation of policy also enables an agent to express the policy under which it is operating. It is easy to transmit, say, a policy message outlining the level of security required for any of several possible upcoming conversations for which the recipient already has the specifications. In contrast, without policy, the "secure" version of each conversation specification needs to be transmitted anew. If two agents agree on a policy at the beginning of a conversation, the amount of communication required to determine a course of action once a violation has occurred can be minimized.

The structure of the conversation depends thus on the nature of the information and how this changes the state of the conversation. By abstracting to the policy level, we enable a set of constraints to support the execution of several conversations, as long as

they have the same *kinds of states* and the same *kinds of transitions*, i.e., the nature of information in a state does not matter as long as there is a common means of mapping input and state to another state in the conversation. If the conversation can be described as a collection of states with transitions between them, then the conversation policy should be describable as a *form* of transition function operating on the current perceived state of the world and the communications the agent is receiving.

This abstraction is powerful because the individual conversation policies can be combined with specifications to create several classes of conversations, all similarly constrained. The constraints the framework imposes are then the conversation policy; and specializations of the conversation policy framework methods are implementations of particular transition functions, which operate on particular classes of conversations. These conversation policies would support transformations by our mechanism, each of which defines a range of possible specializations within the high-level constraints. Radically different behavior between two sets of conversations would imply radically different frameworks, just as the difference between context-free grammars and regular languages implies a greater difference in both the nature of states and the transition function forms of finite automata and stack machines.

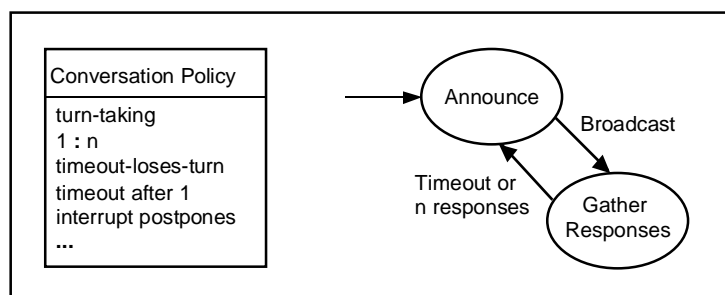


Fig. 2. Policy and specification as seen by the announcer. The policy allows conversations to be postponed, which the conversation specification need not explicitly state.

5 Example

Consider the specification in Figure 2. Agent A_1 , the announcer, broadcasts a message to a group of agents $A_2 \dots A_n$ and gathers responses from the group before continuing. By itself, however, this specification leaves many questions unanswered—for

example, if some agent doesn't respond at all, or responds more than once in a cycle, what should agent A_1 do? These questions may be asked of many specifications, and may have different answers even from one interaction to the next.

The policy in Figure 2 provides answers to some questions of this sort. The policy enforces turn-taking, meaning that agents in the group have only one opportunity to respond to each broadcast. If they do not respond within one minute of each broadcast, they lose the chance to do so during that turn. This might be the case if broadcasts were frequent. If more pressing matters come up during a session, the discussion is postponed (perhaps leaving messages in the announcer's queue to be dealt with later), but it can be expected that the session will resume at some future time.

How might we tailor policies to get usefully different behavior? For policies concerned with fault-tolerance, the same policies could be used in many conversations to handle the same expected problems, but policy can also be used to control conversations during the course of normal interaction as well.

Suppose we combine the specification above with a policy that does not enforce turn taking, but rather says that newer messages from an agent take precedence over older messages. The announcer is forbidden from sharing message data among group members, and the time allowed for responses to each broadcast is 24 hours. Combining the policy and specification with a sales announcer produces a silent auction. If the policy were replaced with one that had a time limit of a few minutes and required the announcer to rebroadcast new information to the group, the same specification could be used to produce an English auction. Using different policies with the same specification as a foundation can produce a variety of desirable behaviors with minimal changes to the agent's code.

6 The Impact of a Policy Mechanism

In this section we discuss the relationship between conversation specifications, policies, and an operational mechanism. We show how policy information can be used to direct the action of an agent without reference to the conversation that agent is having.

Consider a set of state/action pairs with the property that when an agent perceives the world to be in a given state and executes the corresponding action, the world state that results is described by the "state" component of one of the pairs (I/O automata fall conveniently close to this). States with no corresponding actions are end states. Such a set embodies no notion of intent, but an agent can commit to achieving one of the end

states by executing the actions. The point of an action specification is to explicate a series of acts that will result in one of a known set of states.

A conversation specification is such a set of state/action pairs; the specified states are information states and the specified actions are speech acts. A conversation specification explicates a series of speech acts and their triggering states that will result in a one of a known set of information states. An end state may be a goal state, i.e., a state whose achievement is the agent's intent, or a state in which the desired state is known or believed to be either no longer desirable or unachievable.

The conversation specification may specify states and actions that are never realized; e.g., failure-denoting states or error-correcting actions. All actions and states are only partially specified, in the sense that none specify the entire state of the world, because the number of features that might be observed at execution time is infinite, and only a few of these are perceived at specification design time as having any material effect on movement towards the goal.

For example, a plan that includes forming a team might specify neither who is to fill every role on the team, though a specific agent must be cast in each role, nor in what order the roles are to be filled, because the specific order has no effect on the goal state.

Neither the conversation specification nor the policy controls the thread of conversation; the specification specifies the invariant part of the conversation's course, and policy specifies constraints on behavior, not the behavior itself. Control falls to the mechanism that combines the specification object and the policy object to arrive at an executable action at deliberation time. In the remainder of this section, we examine team formation with respect to what is determined by the conversation specification and what is determined by policy.

Assume that an agent is in a state where will listen until it receives a message from another agent. When a message arrives, the agent's policy is to select and commit to achieve one of the end states of a particular conversation specification; in other words, the agent's policy is to have a conversation when contacted. Leaving aside for the moment the question of how the agent makes the selection, assume the agent receives a message asking it to commit to achieving a goal and that it selects a conversation specification wherein it will inform the requester that it has committed if it commits and that it will not commit if it doesn't. This could be a matter of policy; suppose there were many agents available and this was known to the agent. The agent might reason that the best policy would be to report only when it could commit and to keep silent otherwise, in order not to use bandwidth.

Now what happens when an agent achieves a goal to which it has committed? Should the agent report satisfaction to the requester, when there is one? If this were a matter of policy, it could be turned on or off as overarching issues (security, priority, traffic levels, etc.) dictated and overridden as needed by specific policy overrides from the requester.

What should the agent do when it is asked to achieve a goal it believes it cannot achieve by itself? It might be the agent's policy to refuse to commit and to so report. An alternative policy would be to attempt to acquire commitments from other agents to assist. This would begin the team formation phase.

When the agent has acquired commitments from agents whose combined effort it believes can achieve the goal, it builds the team roster of agents $\{A_1, \dots, A_n\}$, marks the team formation goal satisfied, and ends the team formation phase (this ignores the issue of whether everyone on the team must believe the team can achieve the goal in order to commit). It might be the case that the agent must form the team within a given time period; what the agent should do when it does not receive sufficient commitments within the allotted time is a matter of policy. A reasonable policy would be to report to the original requester that the goal is unsatisfiable. This can be enforced at a high level, that is, *whenever* the agent has committed to achieving a goal, and the source of that goal is another agent, the agent must notify the source agent as to the achievement or non-achievement of that goal. The agent holding a team roster for a given goal constructs a joint persistent goal (JPG) (Cohen and Levesque 1991), allocates the subgoals (assume the goal is linearizeable so that allocation is deterministic) and sends each subgoal and the JPG to the appropriate team member. The JPG contains a statement of the original goal and the team roster. When an agent A_i has achieved its subgoal, it multicasts this fact to the rest of the team using the roster in the JPG. Here, policy to notify only the requester must be overridden by JPG-specific policy. Every team member now believes A_i has achieved its subgoal. Once A_i believes that every team member has achieved its subgoal, it believes that the JPG has been satisfied and it multicasts this fact to the rest of the team. At this point, A_i believes that every team member believes that the JPG has been satisfied and is free to leave the team.

7 Conclusions

A conversation policy must be established so that the communicating agents (who may have differing languages) have a common logical and contextual structure for

communicating. This allows each agent to establish predictive models of one another's behavior in response to information and to plan and reason about the outcome of conversations with the other agent. Each agent can establish this model based on information that another agent can perform a certain conversation specification while conforming to certain requirements.

We advocate a separate conversation policy structure that embodies the constraints that will be enforced while a conversation is going on—using a conversation specification as a template or model. A participant in a conversation must have some means of determining whether events that transpire during the conversation bear on the realization of its goals. It is relatively straightforward to specify the normative events in a conversation; the speaker intends to have engendered a specific state in the listener, and the normative response types are limited. On the other hand, it is not generally possible to specify all the exceptions. Even if we could, the necessary responses depend on states of the environment, not states of the conversation. To take the state of the environment into consideration, a policy must be able to constrain the behavior of virtually any conversation specification to which it is applied.

8 Future Developments

It would be useful to define and prove certain formal properties of policies when combined with specifications, for example,

1. Is the question of whether a conversation conforms to a given conversation policy decidable, and if so, how can this be tested?
2. Does conversation X conform to some conversation policy, and if so, which one?
3. What is the maximally confining policy to which a set of conversations conforms?
4. Will the conversation generated from a specification terminate when following a particular policy?
5. Under certain circumstances, a policy may render given specifications impossible. What is the minimal set of constraints that can be established that will still allow a set of conversations to take place?
6. Given a policy that has the potential to render a conversation impossible, what should an agent do?

Consider for a moment the agent as an I/O automaton (IOA) (Lynch and Tuttle, 1989). The IOA's I/O table specifies the agent's behavior. The IOA's input column describes agent's information states. These states can be entered as an agent

internalizes information in messages it has received (i.e., as those messages have perlocutionary effect). The agent then executes the specified internal and external actions specified by the right-hand side of the automaton's I/O table. This formalism has some appeal because it makes a very clear distinction between actions under control of the automaton and those under control of the environment and allows a readable and precise description of concurrent systems.

Analyzing collections of speech acts in terms of I/O automata would be possible if it were not for the dependency of the proofs about the IOAs on their being input-enabled. Agents that filter their stimuli before taking action or replying do not meet this requirement, so the applicability of the IOA theory is questionable.

A formal theory that establishes conversation semantics, describes how the semantics of individual speech acts contribute to conversation, and allows us to demonstrate certain characteristics of combinations of specifications and policies may or may not be useful. When discussing a system whose purpose is to deal with the unexpected, it may be more reasonable to engineer a policy that provides some reasonable capstone when an unanticipated problem arises. Engineering conversations that meet certain requirements, dynamically generating policies and specifications based on beliefs and intentions, and modifying conversations based on changing constraints may allow productive agent behavior even in the absence of a complete theoretical description.

9 In Context

Throughout these papers we see two common issues being addressed: by what means can an agent intend to have, and then have, a conversation, and by what means can an agent manage the process of having conversations in a dynamic environment? Two recurring subproblems are declaring behavioral models for an agent's own use and transmitting these models to other agents; agents need to be able to express the following in both internal and external settings: "This <conversation_spec> is the conversation I want to have" and "This <conversation_policy> is the policy I want to follow." In this paper we have labeled these structures *conversation specifications* and *conversation policies*, respectively.

A primary question roughly separates the papers in this volume into two categories: Are issues of specification and policy to be addressed by a single structural form that unifies specification and policy (Category 1), or by two separate structural forms, one for specification and one for policy, that are somehow composed during the conduct

of a conversation (Category 2)? We are in category 2, having explicitly proposed a *policy object* to be communicated among conversing agents.

An essential question, approached by some authors, but not genuinely disposed of, is: what, exactly, is gained by having two structures? Although efficiency, complexity, and realizeability have been used as motivators, we'd like to see a formal approach that enables decisions of where a particular aspect of discourse should be represented and, in particular, how such decisions are realized when policies and specifications are composed during a conversation.

10 Acknowledgements

This work was performed at Sandia National Laboratories, which is supported by the U.S. Department of Energy under contract DE-AC04-94AL85000.

Regina L. Hunter made numerous valuable comments on the manuscript.

References

1. Bradshaw, J.; Dutfield, S.; Benoit, P.; and Wooley, J. 1997. "KAoS: Toward an Industrial-Strength Open Agent Architecture," in *Software Agents*, AAAI Press/MIT Press.
2. Cohen, P. R., and Levesque, H. J. 1991. Confirmation and Joint Action. In *Proceedings of the 12th Annual International Joint Conference on Artificial Intelligence*, pp 951-959, Menlo Park, CA, Morgan Kaufmann
3. Dennett, D.C. 1987. *The Intentional Stance*. Cambridge, MA: MIT Press.
4. Goldsmith, S.; Phillips, L.; and Spires, S. 1998. A multi-agent system for coordinating international shipping. In *Proceedings of the Workshop on Agent Mediated Electronic Trading (AMET'98)*, in conjunction with *Autonomous Agents '98*, Minneapolis/St. Paul, MN, USA
5. Labrou, Y. and Finin, T 1997. A Proposal for a new KQML Specification, Technical Report, CS-97-03, Dept. of Computer Science and Electrical Engineering, University of Maryland, Baltimore County
6. Lynch, N. A. and Tuttle, M. R. 1989. *An Introduction to Input/Output Automata*, Technical Memo, MIT/LCS/TM-373, Laboratory for Computer Science, Massachusetts Institute of Technology
7. Searle, J. 1969. *Speech Acts*, Cambridge, UK: Cambridge University Press.